Exploring Enterprise, System of Systems, and System and Software Architectures

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Paul Clements
22 January 2009

Software Engineering Institute | Carnegie Mellon

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
22 JAN 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Exploring Enterprise, System of Systems, and System and Software Architectures	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

? As systems grow in complexity in today?s software-intensive world architecture's role becomes crucial at enterprise, system, and software levels? ? In this webinar, we will present our findings from a U.S. Army workshop on architecture that was held at the SEI in September of 2008, under the auspices of the Army Strategic Software Improvement Program (ASSIP). ? We invited accomplished practitioners from government, academia, and industry to discuss the various ?genres? of architecture: enterprise architecture, system-of-systems architecture, system architecture, and software architecture. ? The goal of the workshop was to clarify the relationships among the different genres, explore and identify areas of commonality and difference, and to discuss the role of the Department of Defense Architecture Framework (DoDAF) in helping to capture these architectures.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**47**	

Abstract

- *As systems grow in complexity in today's software-intensive world, architecture's role becomes crucial at enterprise, system, and software levels…*
- *In this webinar, we will present our findings from a U.S. Army workshop on architecture that was held at the SEI in September of 2008, under the auspices of the Army Strategic Software Improvement Program (ASSIP).*
- *We invited accomplished practitioners from government, academia, and industry to discuss the various "genres" of architecture: enterprise architecture, system-of-systems architecture, system architecture, and software architecture.*
- *The goal of the workshop was to clarify the relationships among the different genres, explore and identify areas of commonality and difference, and to discuss the role of the Department of Defense Architecture Framework (DoDAF) in helping to capture these architectures.*

Speaker

- Dr. Paul Clements is a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute, where he has worked since 1994 leading or co-leading projects in software product line engineering and software architecture documentation and analysis.
- He is the co-author of three practitioner-oriented books about software architecture: "Software Architecture in Practice" (1998, second edition 2003), "Evaluating Software Architectures: Methods and Case Studies" (2001), and "Documenting Software Architectures: View and Beyond" (2002). He also co-wrote "Software Product Lines: Practices and Patterns" (2001), and was co-author and editor of "Constructing Superior Software" (1999). In addition, he has also authored dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems. In 2005 and 2006 he spent a year as a visiting faculty member at the Indian Institute of Technology in Mumbai.
- He received a B.S. in mathematical sciences in 1977, and a M.S. in computer science in 1980, both from the University of North Carolina at Chapel Hill. He received a Ph.D. in computer sciences from the University of Texas at Austin in 1994.

Software Engineering Institute

- **Who We Are:** Applied R&D laboratory situated as a college-level unit at Carnegie Mellon University, Pittsburgh, PA, USA

- **Purpose:** Help others make measured improvements in their software engineering practices

- **First objective:** Accelerate the introduction and widespread use of high-payoff software engineering practices and technology identifying, evaluating, and maturing promising or underused technology and practices.

Background

- The SEI held a two-day workshop September 2008, under the auspices of the Army Strategic Software Improvement Program (ASSIP).
- We invited accomplished practitioners from government, academia, and industry to discuss the various “genres” of architecture:
  - enterprise architecture
  - system of systems architecture
  - system architecture
  - software architecture
- The goal of the workshop was to clarify the relationships among the different genres, explore and identify areas of commonality and difference, and to discuss the role of the Department of Defense Architecture Framework (DoDAF) in helping to capture these architectures.

Motivation

- These four genres *each* constitute large areas of research and practice, and *each* play an acknowledged and critical role in building systems and organizations.
- And yet, the various communities do not seem to be talking to each other as much as they might.
- For example, software architects often lament the lateness with which they are brought into system engineering projects.
- *All* of these architecture genres must deal with
  - Satisfaction of functional and quality attribute requirements
  - Evaluation of the architecture for suitability
  - Documentation and communication of the architecture
  - Using the architecture as a blueprint for construction and development

Can we take advantage of this commonality?

Workshop format

- After a selection of opening talks, the workshop dissolved into working groups, each tasked with working on a specific set of issues from the perspective of one of the architecture genres:
  1. What are the major activities involved in each genre?
  2. What is the boundary (e.g., information flow) between architectures in different genres?
  3. What do architectures in each genre need to address in order to be considered successful?
  4. How do we document an architecture in each genre? What notations and approaches are available?
  5. How can DoDAF be used to represent an architecture in each genre? What are its strengths and weaknesses with respect to each genre?

Enterprise architecture

- *"Enterprise architecture is the process of translating business vision and strategy into effective enterprise change by creating, communicating and improving the key requirements, principles and models that describe the enterprise's future state and enable its evolution. The scope of the enterprise architecture includes the people, processes, information and technology of the enterprise, and their relationships to one another and to the external environment. Enterprise architects compose holistic solutions that address the business challenges of the enterprise and support the governance needed to implement them."*

Gartner.

Enterprise Architecture and the Army

- By about 2003, many Army systems were networked, and changes to any system rippled through the network.
- Today, Army systems are connected to inter-agency networks, and change management is even more critical.
- Army enterprise architects must understand the seams, pieces, and boundaries, and take a holistic view of how design decisions in one system affect other systems.

There are all kinds of business

TITLE
Conclusion

Enterprise Architecture is about creating holistic solutions that address a business challenge!

DEPARTMENT	DIRECTORATE					ORGANIZATION
U.S.ARMY	ARCHITECTURE, OPERATIONS & SPACE DIRECTORATE					
	CLASSIFICATION	ISSUED	DATE	SCALE	REVISION	PAGE
	UNCLASSIFIED		1/22/2009	1 : 1	0708A	12

Enterprise Architecture and the Army

- End users want to connect anywhere and at any time, which leads to concerns about security and scalable service delivery. The Army is addressing this with an enterprise architecture that can be viewed as having three elements:
  - Network Service Center
  - DoD Security and Identity Management
  - Data Strategy

Army EA Focus areas

This is an Army effort to reduce the number of networks and access points, from 400 down to around 5.

This is a joint service effort to support sharing data across all of DoD.

Where do I get what I need, what is the format, who has access to it? This includes the unsolved problem of how to enforce need-to-know.

System of Systems (SoS) Architecture

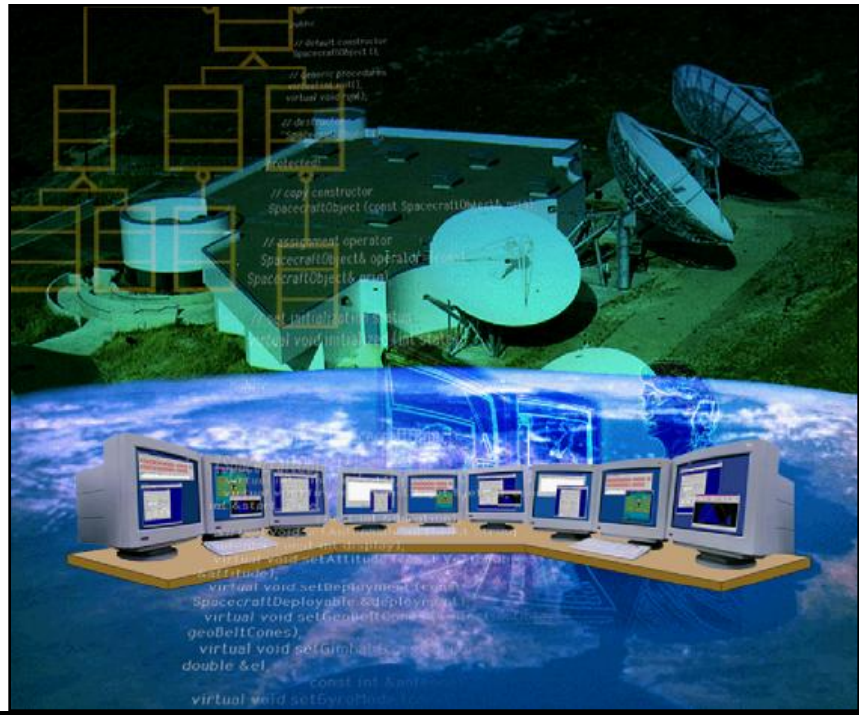
- A SoS is a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities.
- Varieties:
  - **Directed:** SoS objectives, management, funding and authority in place; systems are subordinated to the SoS
  - **Acknowledged:** SoS objectives, management, funding and authority in place; systems retain their own management, funding and authority in parallel with the SoS
  - **Collaborative:** No objectives, management, authority, responsibility, or funding at the SoS level; systems voluntarily work together to address shared or common interest
  - **Virtual:** Like collaborative, but systems don't know about each other (for example, the Internet)

Acknowledged SoS Programs

- Within the DoD, Acknowledged SoS programs are the most common.
  - They are typically an ensemble of individual existing systems brought together to satisfy user capability needs.
  - They are typically not new acquisition efforts.
  - The SoS manager does not control requirements or funding of the individual systems and therefore likely must rely on influencing skills rather than directing skills.
  - The focus of the SoS is on evolution of capability over time, which is a necessary consequence of not being able to engineer the SoS from the ground up.
  - SoS capabilities must evolve with the concurrent independent directions and autonomy in the operation and development of constituent systems; thus, multiple levels of objectives, management and technical authorities with independent priorities, funding, and development plans.

System of Systems Architecture

- Developing and evolving a SoS architecture includes
  - creating and maintaining the SoS concept of operations,
  - determining the systems, their functions, and their relationships and dependencies (both internal and external to the SoS),
  - establishing an understanding of the end-to-end functionality, data flows, and communications within the SoS.
- A successful SoS architecture will provide the technical framework for assessing the options and the implications for meeting SoS requirements over time.
- It will have persistence and a tolerance for change.



System Architecture

- *System* - A collection of components organized to accomplish a specific function or set of functions (IEEE 610.12); a collection of components exhibiting emergent properties; big things and small things are "systems."; things with humans inside the boundary and all-machine things are "systems."
- *Architecture, of a system* - A fundamental or unifying structure of a thing (Dictionary); the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution (ANSI/IEEE 1471-2000)*;* a set of information that defines a system's value, cost, and risk sufficiently for the purposes of the system's sponsor (Maier's rule of thumb)
- *Architecture Description* - A collection of products to document an architecture (ANSI/IEEE 1471-2000).
- *Systems Architecting* - The activities of defining, maintaining, improving, and certifying proper implementation of the system architecture.

Mark Maier's Engineering Problem Space

	Simple	◄——►		Complex
Sponsors	One, w/ $	Several, w/ $	One, w/o $	Many, w/o $
Users	Same as sponsors	Aligned with sponsor	Distinct from sponsor	Unknown
Technology	Low	Medium	High	Super-high
Feasibility	Easy	Barely		No
Control	Centralized	Distributed		Virtual
Situation-Objectives	Tame	Discoverable	Ill-structured	Wicked
Quality	Measureable	Semi-measureable		One-shot and unstable

= "textbook"

Mark Maier's System Architecting Characteristics

	Simple	◄——————————►		Complex
Sponsors	One, w/ $	Several, w/ $	One, w/o $	Many, w/o $
Users	Same as sponsors	Aligned with sponsor	Distinct from sponsor	Unknown
Technology	Low	Medium	High	Super-high
Feasibility	Easy	Barely		No
Control	Centralized	Distributed		Virtual
Situation-Objectives	Tame	Discoverable	Ill-structured	Wicked
Quality	Measureable	Semi-measureable		One-shot and unstable

= "classic"

System Architects Must Balance the Tension among…

- **Organization** – Who's doing what? What are they good/bad at? What is their strategic identity?
- **System** – What are we building? What are the components? What are the key technical decisions? How is it tested?
- **Problem** – What are we doing? What delivers value? What is the environment? What is success?
- **Program** – How do we build/operate? Separation of responsibilities.

Maier's advice

THE AEROSPACE CORPORATION

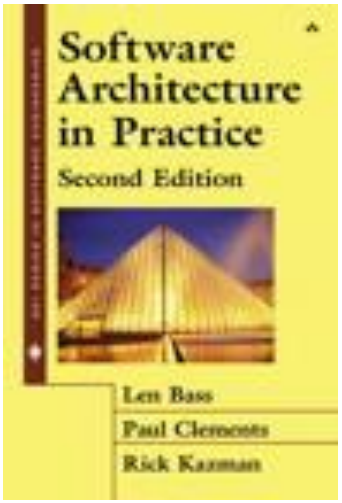
- We engineer (or architect) things (buildings, spacecraft, organizations) - What thing(s) are we working on? If we don't know what the "thing" is, then we are unlikely to be effective at architecting it.
- Architecture is the small set of attributes (structures, rules, protocols) that defines most of the value/cost/risk - Architecture is about discerning the most important from the less important.
- Architecture descriptions should flow from the attributes needed to make a decision - All other definitions (e.g. frameworks) can at best be just generic good practices; they can't define what an architecture is.
- Architecture and Architecture Description standards are different, and satisfy different objectives - Don't confuse them.

Software Architecture

- The quality and longevity of a software-intensive system is largely determined by its architecture.
- Many large system and software failures point to inadequate software architecture education and practices and/or the lack of any real software architecture evaluation early in the life cycle.
- Risk mitigation early in the life cycle has been shown to be a key to averting project failures. The software architecture is an early life cycle artifact and perfectly poised to serve as an early life cycle risk mitigation vehicle. In this way, mid-course correction is possible before great investment.

Software Engineering Institute

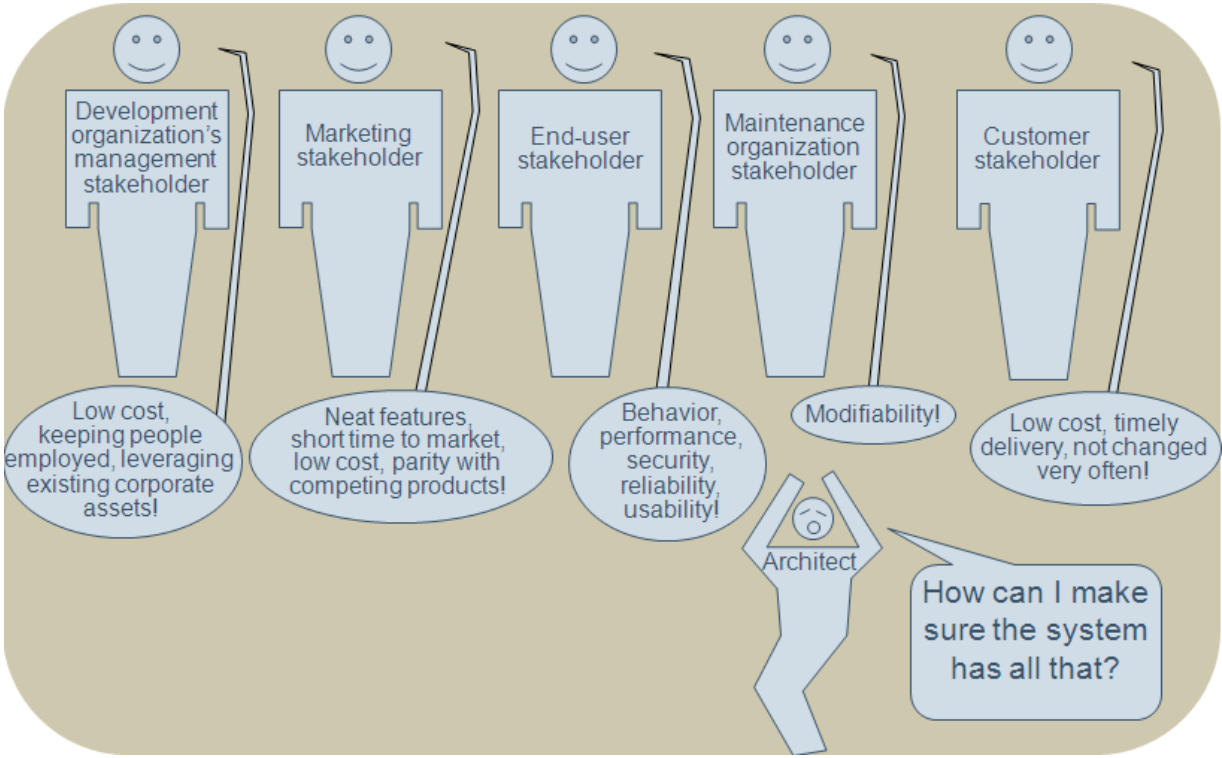
What is software architecture?



- SEI's definition: *The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them.* [Bass et al. 2003]
- The implications of this definition include
  - Software architecture is an abstraction of a system.
  - Software architecture defines the properties of elements.
  - Systems can and do have many structures.
  - Every software-intensive system has an architecture.
  - Having an architecture is different from having an architecture that is known to everyone.
- If you don't develop an architecture, you will get one anyway – and you might not like what you get!

Role of software architecture

- Software architecture is the primary carrier of quality attributes.
- Quality attributes often reflect user needs. For example: required capability, low learning threshold, ease of use, predictable behavior, dependable service, timely response, timely throughput, protection from unintended intruders and viruses…
- Quality attributes also represent the needs of other stakeholders as well.

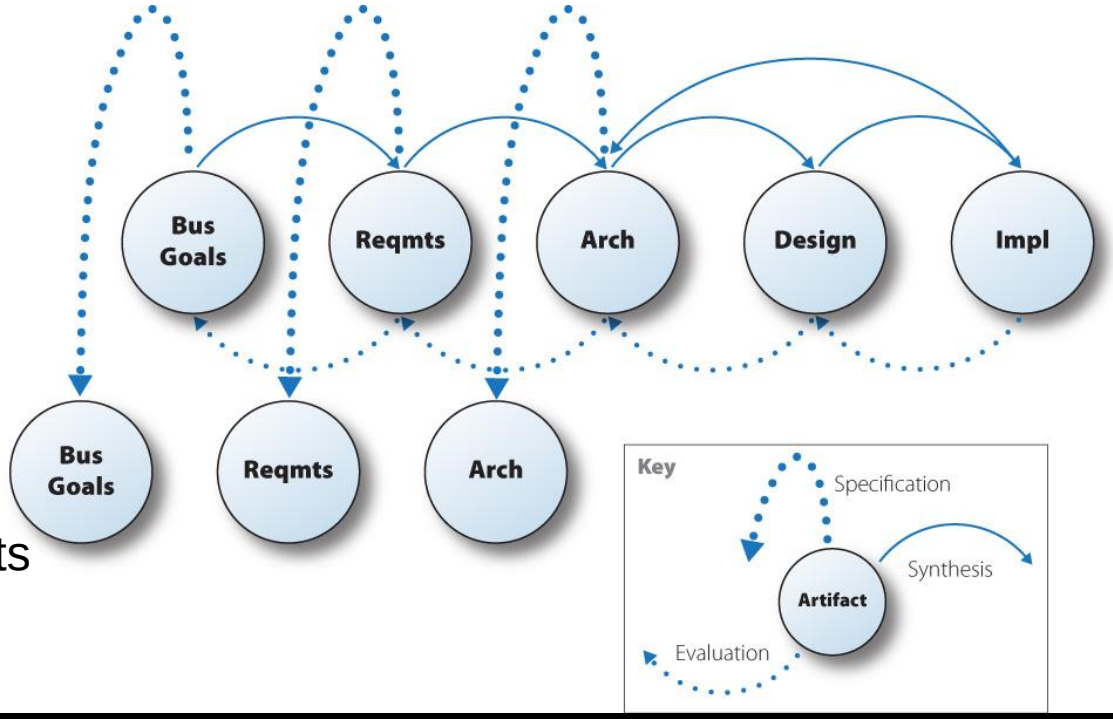


Software architecture and the other genres

- Software architectures “show up” in the architectures of the other genres.
  - Enterprise architecture and system architecture provide an environment in which software lives.
  - Both provide requirements and constraints to which software architecture must adhere.
  - Elements of both are likely to contain software architecture, but neither substitutes for or obviates a software architecture.
  - Both software and system architectures are critical for ensuring that the system meets its business and mission goals.
  - Each system in a SoS has system and software architectures.

Activities in software-architecture-centric development

- creating the business case for the system
- understanding the requirements
- creating and/or selecting the architecture
- documenting/communicating the architecture
- evolving the architecture so that it continues to meet business and mission goals
- analyzing or evaluating the architecture
- setting up the appropriate tests and measures against the architecture
- implementing the system based on the architecture
- ensuring that the implementation conforms to the architecture



Major Activities Involved in Each Genre

- Activities identified by the working groups fall into four major categories:

    1. Understanding goals, context, and requirements
    2. Architecture creation, evaluation, documentation
    3. Managing the architecture post-creation
    4. Assisting in post-architecture activities

Activity #1: Understanding goals, context, and requirements -1

- **EA**
  - Aligning the architecture with other infrastructure decisions within the enterprise
  - Defining architecturally significant requirements
  - Modeling the “as-is” current state architecture
- **SoS**
  - Decomposing objectives into a set of high-level SoS and system requirements.
  - Determining the applicable measures of performance and measures of effectiveness (MOPs and MOEs) (e.g., quality attributes)
- **SoS** (continued)
  - Determining the capability objectives of the SoS.
  - Understanding the architecturally significant aspects of the SoS.
  - Understand to whom the systems belong as well as their positions in their relative development cycles.
  - Understanding the concept of operations (CONOPS) for the SoS.
  - Understanding the context or environment in which the SoS will operate
  - Understanding the vignettes and associated mission threads that describe the dynamics of the SoS.

Activity #1: Understanding goals, context, and requirements -2

- **SW**
  - Creating the business case for the system
  - Establish quality attribute requirements
  - Requirements analysis to produce ASRs
  - Stakeholder and concern analysis
  - Understanding the requirements

- **SYS**
  - Forming system concept; developing a CONOPS and a mission concept

Activity #2: Architecture creation, evaluation, documentation

- **EA**
  - Defining a future state that is aligned with the enterprise business/mission goals
  - Designing, documenting, and evaluating the architecture
- **SoS**
  - Decide which functional elements within the SoS will meet the capability objectives.
  - Develop and document the architecture.
- **SW**
  - Analyzing or evaluating the architecture
  - Creating and/or selecting the architecture
- **SW** (continued)
  - Documenting and communicating the architecture
  - Establishing traceability between architecture and requirements
  - Evaluating
  - Figuring out a representation (viewpoints)
  - Filling in the views
- **SYS**
  - Architecture evaluation
  - Determining and structuring the problem the system is to address.
  - Development of system architecture.
  - Documenting and communicating the system architecture.

Activity #3: Managing the architecture post-creation

- **EA**
  - Governance of the evolution of the architecture
- **SoS**
  - Overseeing evolution was an implicit theme
- **SW**
  - Architecture maintenance
  - Evolving the architecture so that it continues to meet business and mission goals
- **SYS**
  - Assistance in validation for use.

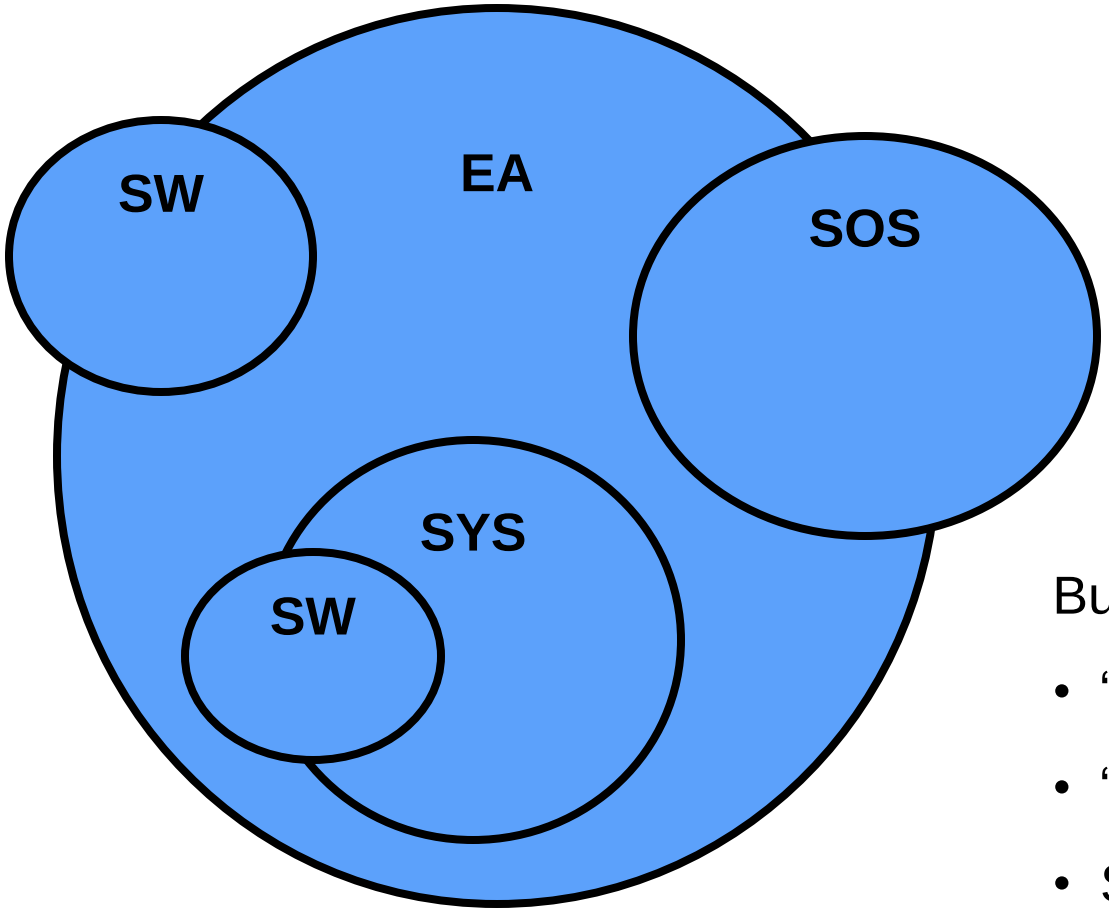
Activity #4: Assisting in post-architecture activities

- **EA**
  - Checking implementation for conformance to the architecture
  - Sustainment of the systems built using the architecture
- **SoS**
  - Determining the high risk activities and how to analyze them.
- **SW**
  - Checking for conformance of downstream artifacts
  - Ensuring that the implementation conforms to the architecture
  - Implementing the system based on the architecture
  - Setting up the appropriate tests and measures against the architecture
- **SYS**
  - Incremental integration strategy.
  - System integrity maintenance.

Why are there no "pre-architecture" activities?

What is the boundary between genres?

- At first, we hoped for some “simple” picture to emerge, like this one:



?

But what does nesting mean?

- “Is a subset of” -- no!
- “Requirements derived from”
- Something else?

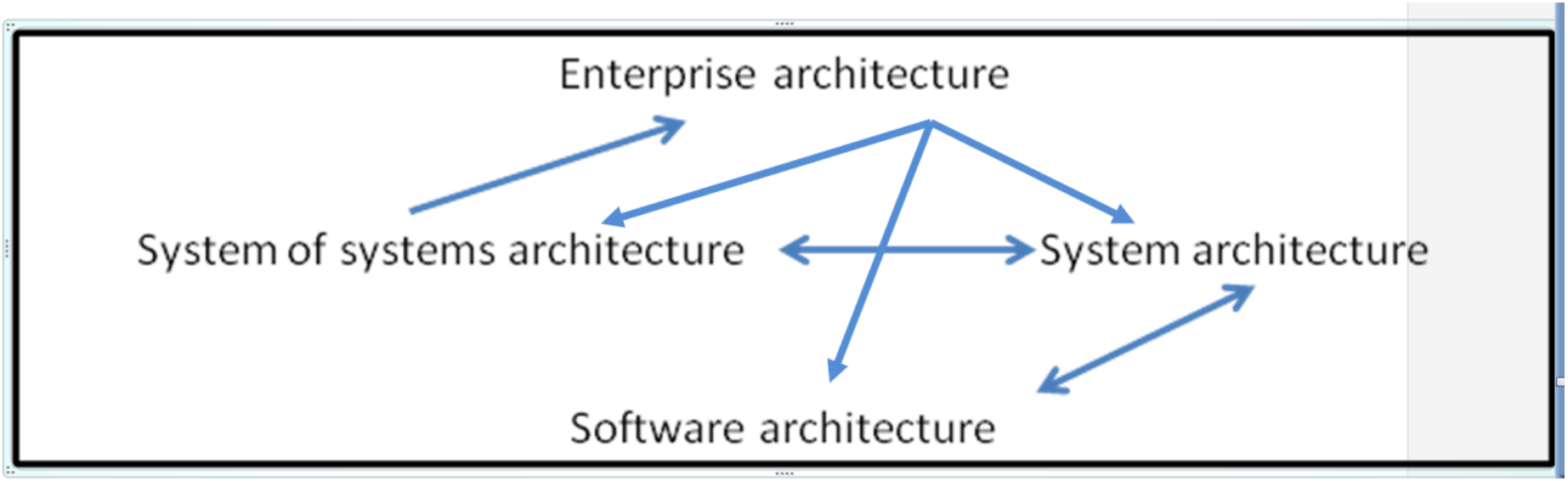
What is the boundary between genres?

- **EA:** The essential tasks of an enterprise change slowly over time, however the supporting technology changes rapidly. Current technology capabilities have a significant impact on enterprise goals, which are then expressed as quality attributes which drive the Enterprise Architecture. Software Architectures, System Architectures, and Systems of Systems Architectures are constrained by the Enterprise Architecture and at the same time are enablers of the Enterprise Architecture.
- **SoS:** The boundaries of SoS architecture tend to be at the system architecture level at the low end and at the enterprise architecture level at the high end. The architecture for an Acknowledged SoS is an 'overlay' on existing systems that have their own architectures. At the higher level, SoS exist within one or more enterprises. Consequently, those SoS must address the tenets and/or constraints of the respective enterprise architectures.

What is the boundary between genres?

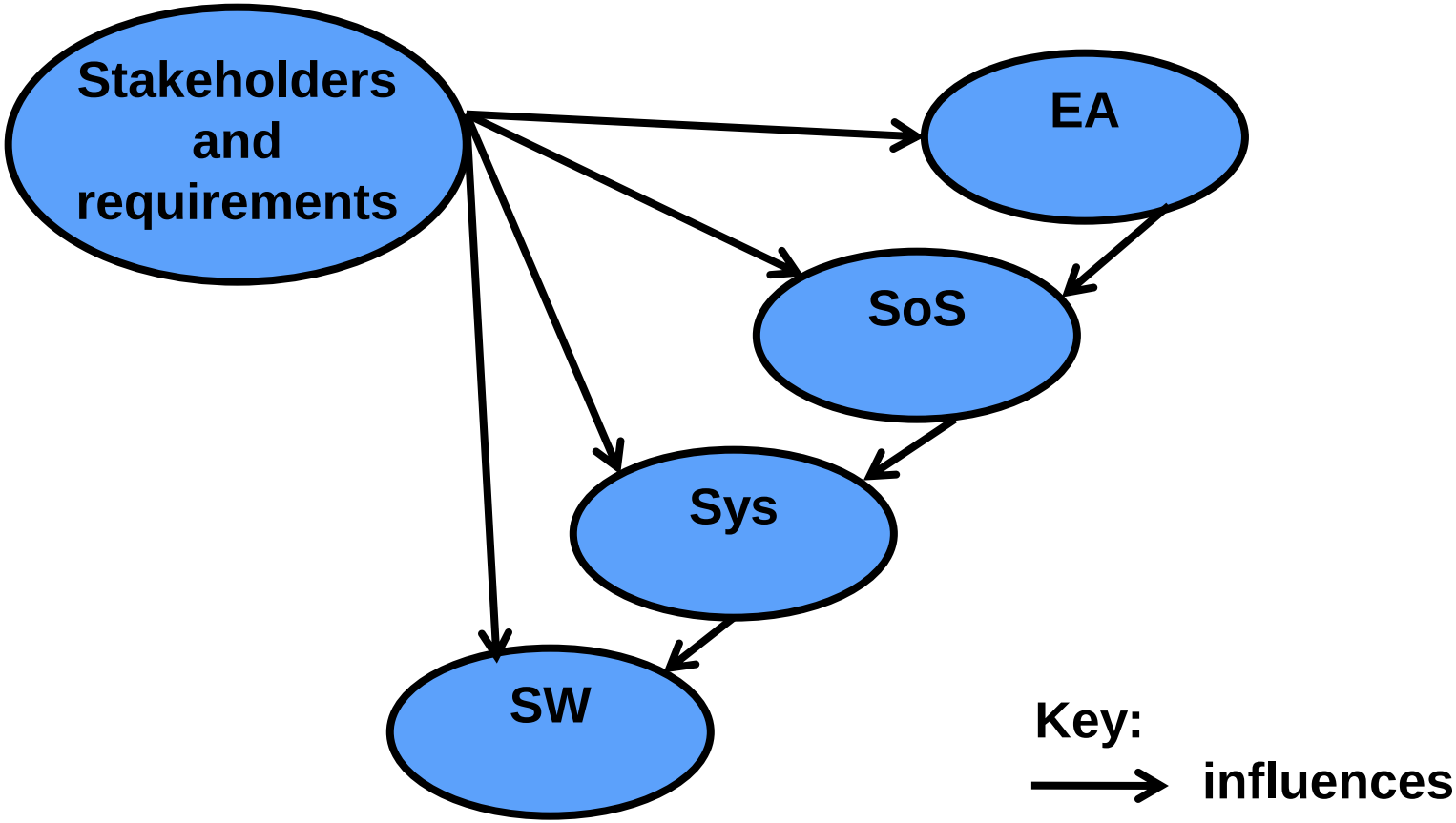
- **SYS**: If an SoS is directive about components, then SoS drives system architecture, however, in the case of using legacy system components, the legacy system architecture will drive the SoS architecture. Also, for a system that is in a product line environment, the software architecture will ride above the system architecture in a sense; while at other times the software architecture will be subsumed within the system architecture.
- **SW**: The primary "interface" is the system/software boundary. Information flow here includes the areas of business and technical risk, the scope of the system and its boundaries, and the pedigree of the requirements: what's a constraint, what's a hard requirement, what's simply a desirable goal, etc. Specific information needs depend on the system and its context. The group felt that in most cases, the software architect is on the receiving end of performance and functionality requirements and little else.

What is the boundary between genres?



- An arrow from one genre to another means that the working group associated with the arrow's tail strongly mentioned the genre associated with the arrow's head.

What is the boundary between genres?



Criteria for success

Genre	Criteria for success
EA	Describe the current processes and workflows that the enterprise uses to achieve its business/mission goals, and must support quantifying the extent to which the enterprise is achieving those goals. The architecture must have evolution options that allow the enterprise to change processes and workflows to meet new business/mission goals. Evolution must be "resource informed", that is, the evolution plan must fit within the funding, people, expertise, and capital constraints of the enterprise.
SoS	Allowing constituent systems as much autonomy as possible. Ability to accommodate changes in systems, while limiting the impacts of change on other parts of the SoS. Ability to accommodate those systems that cannot change Ability to establish a degree of understanding (e.g., an SLA) between the SoS and its systems.
SYS	Ability to achieve mission success. In an "uncertain requirements" environment, then incremental architecture development approach is necessary
SW	The usual list of quality attributes, making sure to include "implementability"

How do we document an architecture in each genre?

Genre	Approaches for capturing architecture
EA	No "one size fits all", no de facto standards. Frameworks like Zachman offer a starting point. FEA offers guidelines for development, adoption, and institutionalization. Scale is important – tooling must support scale, and choice of tooling may impact documentation approach.
SoS	No standard approach. Some SoS programs produce architecture documents, but many forego specific architecture documentation, opting instead to develop white papers with rationale on important aspects of the SoS architecture (such as performance, fault tolerance, or security). Still other programs have attempted to use integrated databases containing requirements, schedules, allocation responsibilities, budgets, etc. Various commercial tools, while usually not specifically developed for use at a SoS level, are often applied and adapted to the needs of SoS architects.
SYS	Usual approaches include block diagrams, use cases, context diagrams and versions of DoDAF (sequence and event traces); value and objective models (text and graphs – value curves); and prototyping, simulation and analysis reports. Capturing and documenting the quality attribute requirements and how the system architecture supports them is an area that needs additional work. The transition between system and software architecture views also could use some attention.
SW	Standard approaches include Kruchten's (later Rational's) 4+1 Views approach, SEI's "View and Beyond" approach , and IEEE 1471 approach

How does DoDAF help?

- There was a clear consensus that DoDAF is helpful in some areas, but is neither necessary nor sufficient to capture a high-quality rendition of an architecture in any genre.

Genre	DoDAF helps	DoDAF doesn't help
EA	"Fit for purpose" philosophy of 2.0 may help.	Standardization of required views and view representations may be helpful.
SoS	Useful for some views	Composability of views Automated analysis 2.0 doesn't appear to add value
SYS	Good basis for dialog about architecture	Tends to be used as post-design tool only Little support for analysis Cost models Discipline-specific models Quality attribute specifications Schedule Software interfaces User interfaces Interface design Layering abstractions Cross-cutting system quality attributes Poor support for back-end analysis Provides little to no value added compared to other current practices.
SW	SV5: might be the starting point for a logical view. OV2 and OV3: information exchange is covered. AV1 and OV1: provide contextual views, and those are useful for software OV7 and SV11: logical data model and implementation of the data model	Module (build-time) views DoDAF not sufficient to represent software architecture

A Recurring Theme

- Getting architects involved earlier and in more depth!

Next steps

- Nail down the boundaries across genres
- Can we create a 1471-style standard for architecture description that would apply to all genres?
- Can we create a “genre-independent” architecture evaluation method?
  - SEI is working on System ATAM and starting to work on an evaluation approach for SoS
- Understand which business and organizational goals lead to architectural requirements (and which do not)

Software Architecture Courses & Conference from the SEI

Upcoming Architecture Courses:

- **Software Architecture: Principles and Practices:** http://www.sei.cmu.edu/products/courses/saf.html
  - March 24-25, 2009 (SEI Pittsburgh, PA)
- **Software Architecture Design and Analysis:** http://www.sei.cmu.edu/products/courses/saad.html
  - April 1-2, 2009 (SEI Pittsburgh, PA)
- **Documenting Software Architectures:** http://www.sei.cmu.edu/products/courses/dsa.html
  - May 20-21, 2009 (SEI Pittsburgh, PA)

Upcoming Conference:

- **Fifth SEI Architecture Technology User Network (SATURN) Conference** http://www.sei.cmu.edu/architecture/saturn/2009/
  - May 4-7, 2009 | Radisson Greentree | Pittsburgh, Pennsylvania

SATURN 2009

5th anniversary SATURN

Are you interested in learning more? Visit http://www.sei.cmu.edu/architecture/saturn/ to

- SATURN → TECHNOLOGY: Find out about the SEI software architecture work, current research, tools and practices, news, and how the SEI can help you.
- SATURN → NETWORK: Stay connected to architecture experts through the SATURN Network on LinkedIn.
- SATURN 2009 (5th anniversary SATURN): Attend SATURN 2009 – the annual conference that brings together experts from around the world to exchange best practices in developing, acquiring, and maintaining software, systems, and enterprise architecture. Registration now open!

Exploring genres of architecture at SATURN

- Keynote Address: John Zachman
- Tutorial: *Embedded Systems Engineering with the AADL*: Modeling & Analysis, by John Hudak, David Gluch, and Bruce Lewis
- Talks
  - *Bridging Systems and Software Architecture* . Anne-Marie Buibish, James Lewis, Elizabeth Penisten, Amy Lange and Caleb Conley.
  - *A Simple and Flexible Specification for an Enterprise Architecture Practice.* David Cuyler.
  - *Limits to the Use of the Zachman Framework in Developing and Evolving Architectures for Complex Systems of Systems* . Suzanne Garcia and Philip Boxer.
  - *Architecting Your Organization* . Kenneth Kunkel.
  - *Career Track for Architects in IT Service Provider Organizations* . Shankar Kambhampaty.
  - *The Role and Development of an Enterprise Architect: A Devil Advocate's Perspective* . Robert Ellinger.

Contact Information and Resources

- **Contact Information**

- **Dr. Paul C. Clements**
- Software Engineering Institute
- Carnegie Mellon University
- Pittsburgh, PA 15213

- Email: clements@sei.cmu.edu

- **World Wide Web:**
- www.sei.cmu.edu/architecture

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

Software Engineering Institute | Carnegie Mellon